УДК 621.391
DOI: 10.18799/29495407/2025/2/88
Шифр специальности ВАК: 2.2.4

Пример псевдопараллельного выполнения трёх задач на микроконтроллере STM32F103 без использования операционной системы

В.А. Вдовин, И.В. Трубин, В.Г. Трубин✉, П.Д. Шендрик

Новосибирский Государственный Технический Университет, Россия, г. Новосибирск

✉trubin@corp.nstu.ru

Аннотация. Представлен пример псевдопараллельного выполнения трёх задач на микроконтроллере STM32F103 без использования операционной системы. Описываются процессы, включающие мигание встроенным светодиодом, передачу и получение данных через последовательный интерфейс USART1 и смену частоты мигания встроенного светодиода по зажатию кнопки с обработкой дребезга контактов. Программная часть организована с использованием конечных автоматов, что позволяет достигнуть псевдомногозадачности в системе. Статья подробно рассматривает инициализацию портов ввода–вывода, настройку последовательного интерфейса USART и таймера.

Ключевые слова: автоматное программирование, микроконтроллер STM32F103, псевдопараллельное выполнение, псевдомногозадачность, конечные автоматы, USART, управление светодиодом, обработка дребезга контактов кнопки, инициализация, прерывание, программирование на языке Си

Для цитирования: Пример псевдопараллельного выполнения трёх задач на микроконтроллере STM32F103 без использования операционной системы / В.А. Вдовин, И.В. Трубин, В.Г. Трубин, П.Д. Шендрик // Известия Томского политехнического университета. Промышленная кибернетика. – 2025. – Т. 3. – № 2. – С. 1–7. DOI: 10.18799/29495407/2025/2/88

UDC 621.391
DOI: 10.18799/29495407/2025/2/88

Example of pseudo-parallel execution of three tasks on an STM32F103 microcontroller without using an operating system

V.A. Vdovin, I.V. Trubin, V.G. Trubin✉, P.D. Shendrik

Novosibirsk State Technical University, Novosibirsk, Russian Federation

✉trubin@corp.nstu.ru

Abstract. This article provides an example of pseudo-parallel execution of three tasks on an STM32F103 microcontroller without using an operating system. The processes are described, including flashing of the built-in LED, transmitting and receiving data via the serial interface USART1 and changing the frequency of flashing of the built-in LED by pressing the button with the processing of contact rattling. The software part is organized using finite automata, which makes it possible to achieve pseudo-multitasking in the system. The article discusses in detail the initialization of I/O ports, configuring the UART interface and timer, as well as working with interrupts and global variables.

Keywords: automatic programming, STM32F103 microcontroller, pseudo-parallel execution, pseudo-multitasking, finite automata, USART, LED control, processing of button contact rattling, initialization, interrupt, C programming

For citation: Vdovin V.A., Trubin I.V., Trubin V.G., Shendrik P.D. Example of pseudo-parallel execution of three tasks on an STM32F103 microcontroller without using an operating system. *Bulletin of the Tomsk Polytechnic University. Industrial Cybernetics*, 2025, vol. 3, no. 2, pp. 1–7. DOI: 10.18799/29495407/2025/2/88

Введение

Одной из актуальных задач разработки программного обеспечения для микроконтроллеров является создание эффективных и масштабируемых программных решений. В связи с этим часто возникает необходимость в подходах, позволяющих организовать параллельное выполнение задач, избегая при этом сложности многопоточности и взаимодействия между потоками. Автоматное программирование [1] является одним из таких подходов, позволяющих разработчику сосредоточиться на логике работы системы, существенно упростив взаимодействие между задачами, при использовании общих переменных.

Программный код

Рассмотрим автоматное программирование на примере решения типовой задачи на базе микроконтроллера (далее МК) *STM32F103* [2, 3]. Программа должна выполнять следующие задачи:

1) мигать встроенным светодиодом *PC13* с частотой 1 Гц;
2) последовательно выводить данные из заранее созданного массива при получении символа «P» по интерфейсу *USART1* [4, 5];
3) при зажатии кнопки, подключенной к *PA11*, изменять частоту мигания встроенного светодиода *PC13* на 4 Гц. Необходимо обеспечить обработку дребезга контактов кнопки [6].

Всю инициализацию МК выполним в отдельном файле «*main_init.c*».

Блок кода 1. *Файл main_init.c*

```
// Инициализация микроконтроллера
static __INLINE void main_init() {
// Буфер предвыборки разрешён, 2 цикла
задержки для FLASH, см. PM0075
FLASH->ACR = FLASH_ACR_PRFTBE |
FLASH_ACR_LATENCY_2;
// HCLK = SYSCLK
RCC->CFGR |= RCC_CFGR_HPRE_DIV1;
// PCLK1 = SYSCLK / 2 (APB1)
RCC->CFGR |= RCC_CFGR_PPRE1_DIV2;
// PCLK2 = SYSCLK (APB2)
RCC->CFGR |= RCC_CFGR_PPRE2_DIV1;
// Выбираем работу от PLL
RCC->CFGR |= RCC_CFGR_SW_PLL;
// SYSCLK = 72 MHz
RCC->CFGR |= RCC_CFGR_PLLMULL9;
// Тактирования PLL от кварца
RCC->CFGR |= RCC_CFGR_PLLSRC_HSE;
//Запускаем генератор HSE
RCC->CR |= RCC_CR_HSEON;
while (!(RCC->CR & RCC_CR_HSERDY)) {};
// Запускаем PLL
RCC->CR |= RCC_CR_PLLON;
while ((RCC->CR & RCC_CR_PLLRDY) == 0) {};
// Ждём когда PLL начнёт тактирование
while ((RCC->CFGR & RCC_CFGR_SWS) !=
RCC_CFGR_SWS_1) {};
// Отключаем внутренний RC-генератор
//RCC->CR &= ~RCC_CR_HSION;
// JTAG-DP запрещён and SW-DP разрешён
AFIO->MAPR =
(AFIO_MAPR_SWJ_CFG_JTAGDISABLE);
```

Настроим порты ввода–вывода.

Блок кода 2. *Файл main_init.c*

```
// Включаем тактирование подсистем
RCC->APB2ENR = (RCC_APB2ENR_IOPAEN |
RCC_APB2ENR_IOPBEN | RCC_APB2ENR_IOPCEN |
RCC_APB2ENR_AFIOEN);
// PC13 как выход, 10 МГц
GPIOC->CRH &= ~(GPIO_CRH_MODE13 |
GPIO_CRH_CNF13);
GPIOC->CRH |= GPIO_CRH_MODE13_0;
// PA11 как вход с подтяжкой к питанию
GPIOA->CRH &= ~(GPIO_CRH_MODE11 |
GPIO_CRH_CNF11);
GPIOA->CRH |= GPIO_CRH_CNF11_1;
GPIOA->ODR |= GPIO_ODR_ODR11;
```

Выполним настройку последовательного интерфейса *USART1*.

Блок кода 3. *Файл main_init.c*

```
// Настройки по умолчанию:
// 8 информационных бит, 1 стоповый бит,
контроля чётности нет
// Включить тактирование USART1
RCC->APB2ENR |= RCC_APB2ENR_USART1EN;
// PA9 (TX1) AFIO Push-Pull, 10MHz. PA10
(RX1) HiZ, 10MHz
GPIOA->CRH &= ~(GPIO_CRH_MODE9 |
GPIO_CRH_CNF9);
GPIOA->CRH &= ~(GPIO_CRH_MODE10 |
GPIO_CRH_CNF10);
GPIOA->CRH |= (GPIO_CRH_MODE9_0 |
GPIO_CRH_CNF9_1 | GPIO_CRH_CNF10_0);
// Выбор скорости работы порта (9600
бит/с)
// PCLK2 / Baud = 72000000 / 9600 бод
USART1->BRR = 7500;
// Включение USART, передатчика и приемни-
ка
USART1->CR1 = USART_CR1_UE | USART_CR1_TE
| USART_CR1_RE;
```

Настроим таймер-счетчик 2 (*TIM2*) на срабатывание прерывания по переполнению каждую 1 мс.

Блок кода 4. *Файл main_init.c*

```
// Включить тактирование TIM2
RCC->APB1ENR |= RCC_APB1ENR_TIM2EN;
// Режим внутреннего тактирования
TIM2->SMCR &= ~TIM_SMCR_SMS;
// Шаг (квант) счёта 100 мкс
TIM2->PSC = 7200 - 1;
// Переполнение через 1 мс
TIM2->ARR = 10 - 1;
// Разрешить прерывание по переполнению
таймера
TIM2->DIER |= TIM_DIER_UIE;
// Разрешить прерывание в контроллере пре-
рываний
NVIC_EnableIRQ(TIM2_IRQn);
// Включить таймер-счетчик
TIM2->CR1 = TIM_CR1_CEN;
}
```

После настроек рассмотрим главный файл «*main.c*». Сначала необходимо выполнить подключение заголовочных файлов.

Блок кода 5. *Файл main.c*

```
// Описание регистров микроконтроллера
#include "stm32f10x.h"
// Файл с инициализацией микроконтроллера
#include "main_init.c"
```

Для удобства объявим несколько определений.

Блок кода 6. *Файл main.c*

```
// Включить встроенный светодиод PC13
#define LED_PC13_ON GPIOC->BSRR =
GPIO_BSRR_BS13
// Выключить встроенный светодиод PC13
#define LED_PC13_OFF GPIOC->BSRR =
GPIO_BSRR_BR13
// Значение переменной при нажатой кнопке
#define BUTTON_ON 0x00
// Значение переменной при отпущенной
кнопке
#define BUTTON_OFF 0xFF
// Размер массива в байтах
#define ARRAY_BYTES 52
```

Объявим необходимые глобальные переменные.

Блок кода 7. *Файл main.c*

```
// Переменная состояния кнопки
volatile uint8_t BUTTON = 0;
// Счетчик времени для светодиода
volatile uint16_t CT = 0;
// Задержка для светодиода в мс
uint16_t TIMER = 500;
// Состояние автомата №1
uint8_t STATE1 = 0;
// Состояние автомата №2
uint8_t STATE2 = 0;
// Состояние автомата №3
uint8_t STATE3 = 0;
// Массив передаваемых данных (52 байта)
uint8_t ARRAY[ARRAY_BYTES];
// Номер передаваемого байта из массива
uint8_t INDEX;
```

Обработка дребезга контактов кнопки выполнена в прерывании таймера-счетчика *TIM2*, которое вызывается каждую 1 мс. Для того чтобы отделить дребезг контактов кнопки, используем переменную *BUTTON*, которая каждую 1 мс сдвигается влево на один разряд и на место освободившегося бита вписывается текущее состояние кнопки (1 или 0).

Так как в рассматриваемом примере кнопка при нажатии замыкается на «землю» (минус питания), при нажатии на кнопку переменная должна заполнится нулями, а при отпускании кнопки – единицами.

В рамках этого же прерывания происходит уменьшение значения счётчика времени свечения светодиода *CT*, если оно ещё не равна нулю.

Блок кода 8. *Файл main.c*

```
// Прерывание TIM2 (каждую 1 мс)
void TIM2_IRQHandler() {
  // Если это прерывание по переполнению
  if ((TIM2->SR & TIM_SR_UIF) != 0) {
  // Сброс флага переполнения
  TIM2->SR &= ~ TIM_SR_UIF;
  // Наполнение переменной кнопки
  BUTTON <<= 1; if (GPIOA->IDR &
GPIO_IDR_IDR11) BUTTON |= 1;
  // Отсчёт времени
  if (CT > 0) CT--;
  }
}
```

Описание функции *main()* начинается с вызова функции инициализации МК из файла «*main_init.c*», рассмотренного выше, заполнения массива передаваемых данных *ARRAY* прописными и строчными буквами, с помощью кодировки *ASCII* (рис. 1), а также объявления бесконечного цикла *while()*.

DEC	HEX	Символ	DEC	HEX	Символ	DEC	HEX	Символ
56	0x38	8	80	0x50	P	104	0x68	h
57	0x39	9	81	0x51	Q	105	0x69	i
58	0x3A	:	82	0x52	R	106	0x6A	j
59	0x3B	;	83	0x53	S	107	0x6B	k
60	0x3C	<	84	0x54	T	108	0x6C	l
61	0x3D	=	85	0x55	U	109	0x6D	m
62	0x3E	>	86	0x56	V	110	0x6E	n
63	0x3F	?	87	0x57	W	111	0x6F	o
64	0x40	@	88	0x58	X	112	0x70	p
65	0x41	A	89	0x59	Y	113	0x71	q
66	0x42	B	90	0x5A	Z	114	0x72	r
67	0x43	C	91	0x5B	[	115	0x73	s
68	0x44	D	92	0x5C	\	116	0x74	t
69	0x45	E	93	0x5D	]	117	0x75	u
70	0x46	F	94	0x5E	^	118	0x76	v
71	0x47	G	95	0x5F	_	119	0x77	w
72	0x48	H	96	0x60	`	120	0x78	x
73	0x49	I	97	0x61	a	121	0x79	y
74	0x4A	J	98	0x62	b	122	0x7A	z
75	0x4B	K	99	0x63	c	123	0x7B	{
76	0x4C	L	100	0x64	d	124	0x7C	|
77	0x4D	M	101	0x65	e	125	0x7D	}
78	0x4E	N	102	0x66	f	126	0x7E	~
79	0x4F	O	103	0x67	g			

Рис. 1. *Кодировка ASCII (коды 56–126)*
Fig. 1. *ASCII encoding (codes 56–126)*

Блок кода 9. *Файл main.c*

```
int main(void) {
  // Инициализация МК
  main_init();
  // Заполнение массива данных символами
  for (uint8_t i = 65; i <= 90; i++){
  ARRAY[i-65] = i;
  ARRAY[i-39] = i+32;
  }
  while (1) {
```

Далее идёт описание конечных автоматов, которые и выполняют поставленные задачи.

Для мигания светодиодом *PC13* используется переменная-счётчик *CT*, в которую можно записать значение времени в мс (например, 500) и ожидать момента, пока эта переменная не станет нулевой, что и будет означать окончание отсчета выставленного времени.

Блок кода 10. *Файл main.c*

```
// ======== Конечный автомат №1 ========
// Мигание встроенным светодиодом PC13
switch(STATE1) {
  case 0:
  if (CT == 0) {
  LED_PC13_ON;
  CT = TIMER;
  STATE1 = 1;
  }
  break;
  case 1:
  if (CT == 0) {
  LED_PC13_OFF;
  CT = TIMER;
  STATE1 = 0;
  }
  break;
}
```

Второй конечный автомат занимается последовательной отправкой данных из массива по интерфейсу *USART1*, если принят символ «P».

Блок кода 11. *Файл main.c*

```
// ======== Конечный автомат №2 ========
// Вывод данных из массива по команде «P»
через USART1
switch(STATE2) {
  case 0:
  // Если пришёл символ по USART
  if (USART1->SR & USART_SR_RXNE) {
  // И символ равен 'P'
  if (USART1->DR == 'P') {
  STATE2 = 1;
  INDEX = 0;
  }
  }
  break;
  case 1:
  // Если передатчик готов к передаче
  if (USART1->SR & USART_SR_TXE) {
  USART1->DR = ARRAY[INDEX++];
  // Если это был последний байт
  if (INDEX >= ARRAY_BYTES)
  STATE2 = 0;
  }
  break;
}
```

Третий конечный автомат выполняет переключение частоты мигания встроенного светодиода *PC13* при зажатии кнопки. Зажатие кнопки определяется по переменной *BUTTON*, значение которой меняется через прерывание *TIM2*, описанное в блоке кода 8.

Блок кода 12. *Файл main.c*

```
// ======== Конечный автомат №3 ========
// Переключение частоты мигания встроенно-
го светодиода PC13 по кнопке PA11
switch(STATE3) {
  case 0:
  if (BUTTON == BUTTON_ON) {
  TIMER = 125;
  STATE3 = 1;
  }
  break;
  case 1:
  if (BUTTON == BUTTON_OFF) {
  TIMER = 500;
  STATE3 = 0;
  }
  break;
}
  }
}
```

На этом описание цикла *while() и* функции *main()* завершены.

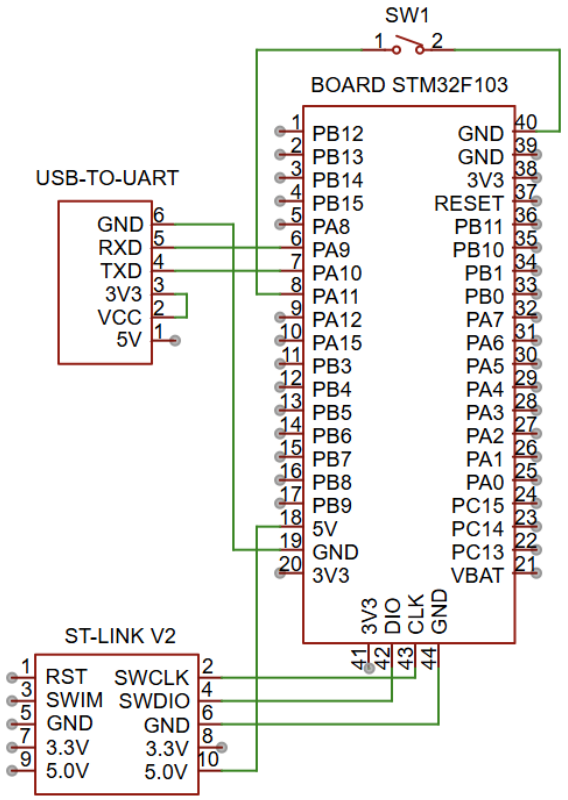

Рис. 2. *Схема подключений*
Fig. 2. *Connection scheme*

Сборка схемы и проверка кода

Для реализации поставленной задачи предлагается взять *TTL* преобразователь *USB-TO-UART* на основе микросхемы *CH340* [7], программу для работы с виртуальными последовательными портами (так называемыми *COM*-портами) *Terminal v1.9b*, кнопку, которую можно имитировать обычным замыканием контактов и программатор *ST-LINK V2*.

При отсутствии *ST-LINK V2* можно воспользоваться технологией программирования микроконтроллеров с помощью встроенного загрузчика по *USART* [8].

Схема подключений изображена на рис. 2.

После сборки схемы (рис. 3) и программирования МК можно будет увидеть мигание встроенного светодиода *PC13* с частотой 1 Гц, а при зажатии кнопки – 4 Гц. В то же время при отправке символа «P» (с помощью программы *Terminal*) на экран будут выведены пятьдесят два символа. При этом отправка символов компьютеру никак не скажется на других задачах.

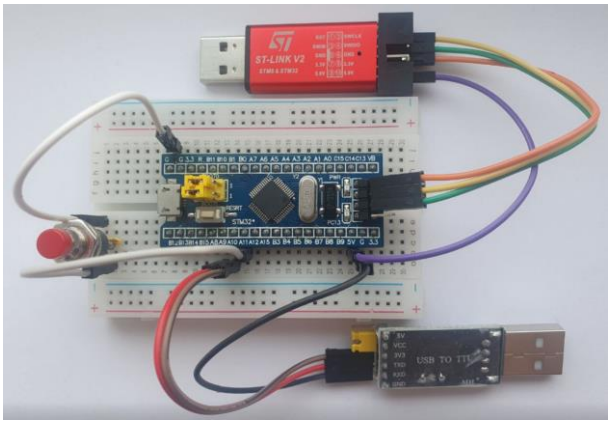

Рис. 3. *Фото собранной схемы*
Fig. 3. *Photo of the assembled circuit*

Данные, полученные и отправленные по последовательному интерфейсу *USART1*, изображены на рис. 4.

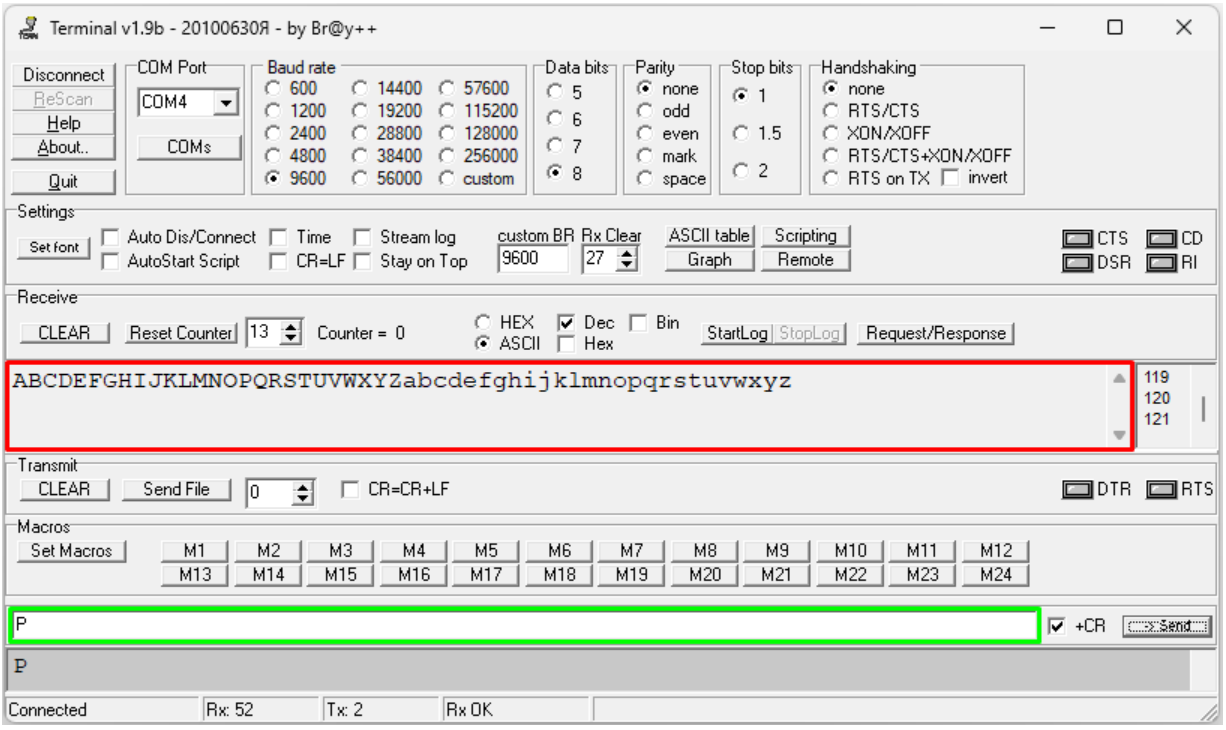

Рис. 4. *Результат обмена данными в программе Terminal (окна приёма и передачи данных)*
Fig. 4. *Result of data exchange in the Terminal program (windows for receiving and transmitting data)*

Заключение

- Предложен способ простой реализации псевдомногозадачности на примере микроконтроллера *STM32F103* без использования операционной системы.
- Приведен и разобран пример кода на языке программирования Си, позволяющий выполнять мигание встроенным светодиодом *PC13*, обмен данными по *USART1* и переключение частоты мигания светодиода при зажатии кнопки, с обработкой дребезга контактов.
- Показано, что применение автоматного программирования позволяет значительно упростить процесс разработки программного обеспечения для микроконтроллеров. Использование конечных автоматов способствует более яс-

ной организации кода и логики работы программы.
- Реализация псевдомногозадачности с помощью конечных автоматов позволяет существенно упростить взаимодействие между задачами при использовании общих переменных. Это особенно важно для встраиваемых систем, где ресурсы ограничены.

СПИСОК ЛИТЕРАТУРЫ

1. Введение в автоматное программирование микроконтроллеров STM32 / В.А. Вдовин, И.В. Трубин, В.Г. Трубин, П.Д. Шендрик // Известия Томского политехнического университета. Промышленная кибернетика. – 2024. – Т. 2. – № 4. – С. 21–25. DOI: 10.18799/29495407/2024/4/75 EDN: RJPJXI
2. Документация на микроконтроллер STM32F103. URL: https://www.st.com/resource/en/datasheet/stm32f103c8.pdf (дата обращения: 05.11.2024).
3. Reference manual STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced Arm®-based 32-bit MCUs. URL: https://www.st.com/content/ccc/resource/technical/document/reference_manual/59/b9/ba/7f/11/af/43/d5/CD00171190.pdf/files/CD00171190.pdf/jcr:content/translations/en.CD00171190.pdf (дата обращения: 05.11.2024).
4. Близнюк А.Е., Трубин В.Г., Саблина Г.В. Представление и отображение информации при работе с микроконтроллерами STM32 // Автоматика и программная инженерия. – 2022. – № 3 (41). – С. 60–78. EDN: GGMTQO
5. Жмудь В.А., Трубин И.В., Трубин М.В. Обмен данными между компьютером и микроконтроллером STM32F100 по последовательному интерфейсу связи RS-232 // Автоматика и программная инженерия. – 2015. – № 1 (11). – С. 45–51. EDN: VJOEZL
6. Незванов А.И., Трубин В.Г. Подключение кнопок, дискретных сигналов к микроконтроллеру STM32F103 // Автоматика и программная инженерия. – 2018. – № 4 (26). – С. 34–43. EDN: YRIIUX
7. USB to Serial Port Chip CH340. URL: https://www.wch-ic.com/products/CH340.html (дата обращения: 26.01.2025).
8. Программирование микроконтроллеров STM32F10x с помощью встроенного загрузчика по USART / А.Е. Близнюк, В.А. Жмудь, М.В. Трубин, В.Г. Трубин // Программная инженерия. – 2022. – Т. 13. – № 3. – С. 132–141. DOI: 10.17587/prin.13.132-141 EDN: OJTMKC

Информация об авторах

Владислав Андреевич Вдовин, студент, кафедра автоматики, Новосибирский государственный технический университет, Россия, 630073, г. Новосибирск, пр. Карла Маркса, 20; vladislickness@mail.ru

Игорь Витальевич Трубин, старший преподаватель, кафедра защиты информации, Новосибирский государственный технический университет, Россия, 630073, г. Новосибирск, пр. Карла Маркса, 20; i.trubin@corp.nstu.ru; https://orcid.org/0009-0001-1466-6001

Виталий Геннадьевич Трубин, старший преподаватель, кафедра автоматики, Новосибирский государственный технический университет, Россия, 630073, г. Новосибирск, пр. Карла Маркса, 20; trubin@corp.nstu.ru

Павел Дмитриевич Шендрик, студент, кафедра автоматики, Новосибирский государственный технический университет, Россия, 630073, г. Новосибирск, пр. Карла Маркса, 20; pasha.shendrick@yandex.ru

Поступила: 20.02.2025
Принята: 23.05.2025
Опубликована: 27.06.2025

REFERENCES

1. Vdovin V.A., Trubin I.V., Trubin V.G., Shendrik P.D. Introduction to automatic programming of STM32 microcontrollers. *Bulletin of the Tomsk Polytechnic University. Industrial cybernetics*, 2024, vol. 2, no. 4, pp. 21–25. (In Russ.) DOI:10.18799/29495407/2024/4/75 EDN: RJPJXI
2. *Documentation for the STM32F103 microcontroller*. (In Russ.) Available at: https://www.st.com/resource/en/datasheet/stm32f103c8.pdf (accessed: 5 November 2024).
3. *Reference manual STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced Arm®-based 32-bit MCUs*. Available at: https://www.st.com/content/ccc/resource/technical/document/reference_manual/59/b9/ba/7f/11/af/43/d5/CD00171190.pdf/files/CD00171190.pdf/jcr:content/translations/en.CD00171190.pdf (accessed: 5 November 2024).
4. Bliznyuk A.E., Trubin V.G., Sablina G.V. Representation and display of information when working with STM32 microcontrollers. *Automatics & Software Enginery*, 2022, no. 3 (41), pp. 60–78. (In Russ.) EDN: GGMTQO
5. Zhmud V.A., Trubin I.V., Trubin M.V. Exchange of data between the computer and the microcontroller STM32F100 by serial communication interface RS-232. *Automatics & Software Enginery*, 2015, no. 1 (11), pp. 45–51. (In Russ.) EDN: VJOEZL
6. Nezvanov A.I., Trubin V.G. Connecting buttons, discrete signals to the STM32F103 microcontroller. *Automatics & Software Enginery*, 2018, no. 4 (26), pp. 34–43. (In Russ.) EDN: YRIIUX
7. *USB to Serial Port Chip CH340*. Available at: https://www.wch-ic.com/products/CH340.html (accessed: 26 January 2025).
8. Bliznyuk A.E., Zhmud V.A., Trubin M.V., Trubin V.G. Programming of STM32F10x microcontrollers using the built-in USART bootloader. *Software Engineering*, 2022, vol. 13, no. 3, pp. 132–141. (In Russ.) DOI: 10.17587/prin.13.132-141 EDN: OJTMKC

Information about the authors
Vladislav A. Vdovin, Student, Novosibirsk State Technical University, 20, Karl Marks avenue, Novosibirsk, 630073, Russian Federation; vladislickness@mail.ru
Igor V. Trubin, Senior Lecturer, Novosibirsk State Technical University, 20, Karl Marks avenue, Novosibirsk, 630073, Russian Federation; i.trubin@corp.nstu.ru; https://orcid.org/0009-0001-1466-6001
Vitaly G. Trubin, Senior Lecturer, Novosibirsk State Technical University, 20, Karl Marks avenue, Novosibirsk, 630073, Russian Federation; trubin@corp.nstu.ru
Pavel D. Shendrik, Student, Novosibirsk State Technical University, 20, Karl Marks avenue, Novosibirsk, 630073, Russian Federation; pasha.shendrick@yandex.ru

Received: 20.02.2025
Revised: 23.05.2025
Accepted: 27.06.2025